## Step 3: data Cleaning

### 1: Drop all punctuation and transform all words into lower case.

spooky\_wrd<-unnest\_tokens(spooky,word,text)  
head(spooky\_wrd)

## id author word  
## 1 id26305 EAP this  
## 1.1 id26305 EAP process  
## 1.2 id26305 EAP however  
## 1.3 id26305 EAP afforded  
## 1.4 id26305 EAP me  
## 1.5 id26305 EAP no

### 2: Bi-grams, n-grams

If we wanna get relationships between words, we use n-grams. So far we’ve considered words as individual units, and considered their relationships to sentiments or to documents. However, many interesting text analyses are based on the relationships between words, whether examining which words tend to follow others immediately. we’ll explore some of the methods tidytext offers for calculating and visualizing relationships between words in your text dataset. This includes the token = “ngrams” argument, which tokenizes by pairs of adjacent words rather than by individual ones. We’ll also introduce two new packages: ggraph, which extends ggplot2 to construct network plots, and widyr, which calculates pairwise correlations and distances within a tidy data frame. Together these expand our toolbox for exploring text within the tidy data framework.

#### (1): Tokenizing by n-gram

We’ve been using the unnest\_tokens function to tokenize by word, or sometimes by sentence, which is useful for the kinds of sentiment and frequency analyses we’ve been doing so far. But we can also use the function to tokenize into consecutive sequences of words, called n-grams. By seeing how often word X is followed by word Y, we can then build a model of the relationships between them. We do this by adding the token = “ngrams” option to unnest\_tokens(), and setting n to the number of words we wish to capture in each n-gram. When we set n to 2, we are examining pairs of two consecutive words, often called “bigrams”

# Make a table with one word per row and remove `stop words` (i.e. the common words).  
bigrams<-unnest\_tokens(spooky,bigram, text, token = "ngrams", n = 2)  
head(bigrams)

## id author bigram  
## 1 id00001 MWS idris was  
## 2 id00001 MWS was well  
## 3 id00001 MWS well content  
## 4 id00001 MWS content with  
## 5 id00001 MWS with this  
## 6 id00001 MWS this resolve

bigrams\_HPL<-unnest\_tokens(spooky[spooky$author=='HPL',],bigram, text, token = "ngrams", n = 2)  
head(bigrams\_HPL)

## id author bigram  
## 1 id00002 HPL i was  
## 2 id00002 HPL was faint  
## 3 id00002 HPL faint even  
## 4 id00002 HPL even fainter  
## 5 id00002 HPL fainter than  
## 6 id00002 HPL than the

bigrams\_MWS<-unnest\_tokens(spooky[spooky$author=='MWS',],bigram, text, token = "ngrams", n = 2)  
head(bigrams\_MWS)

## id author bigram  
## 1 id00001 MWS idris was  
## 2 id00001 MWS was well  
## 3 id00001 MWS well content  
## 4 id00001 MWS content with  
## 5 id00001 MWS with this  
## 6 id00001 MWS this resolve

bigrams\_EAP<-unnest\_tokens(spooky[spooky$author=='EAP',],bigram, text, token = "ngrams", n = 2)  
head(bigrams\_EAP)

## id author bigram  
## 1 id00003 EAP above all  
## 2 id00003 EAP all i  
## 3 id00003 EAP i burn  
## 4 id00003 EAP burn to  
## 5 id00003 EAP to know  
## 6 id00003 EAP know the

This data structure is still a variation of the tidy text format. It is structured as one-token-per-row (with extra metadata, such as author, still preserved), but each token now represents a bigram.